A Python implementation for discriminative multivariate data visualization using a Grand Tour of manifold learning nonlinear methods

Angel Astudillo Aguilar

Supervisor: A. Vellido, Ph.D.

Facultat d'Informàtica de Barcelona
UPC

This dissertation is submitted for the degree of
Master of Science
Master in Artificial Intelligence

October 2018

Dedication

I would like to thank God in the first place for all the blessings received in the development of this work. I also thank my beloved wife, Annie Mogrovejo, for being my support and inspiration. My dear father, Freddy Astudillo, who has provided the values I have. My dear Mother, Emma Aguilar, who always inspires me with her history of overcoming. my older brother, Eddy Astudillo, who has always shown me his full support in all facets of my life, up to the present, has also been an example of studies to follow. My second brother, Rommel Astudillo, who has always been responsible to me and I always carry him in my heart. My dear sister, Vanessa Astudillo, to whom I owe a happy childhood and has been one of the reasons for the development of this master.

I also thank UNIVERSIDAD POLITÈCNICA DE CATALUNYA for opening its doors and equipping me with the necessary knowledge to carry out the present work, to all my professors, especially to my supervisor thesis and co-supervisor, who offered the opportunity to develop this work and the necessary help for the realization of this work.

I thank from the deep of my heart also those people that I have not mentioned but who have contributed to achieve this work, it was possible because of you all.

Acknowledgements

And I would like to acknowledge the help from the Ecuadorian government, through the SENESCYT who gave me a scholarship to develop the master degree in artificial intelligence.

Abstract

In this work has been implemented, in python, the manifold grand tour with a GTM structure as a way to improve the intuitiveness of the visualization of the GTM manifold. It is done travelling across each point of the GTM structure and optimizing the discrimination of the classes and increasing the condensation of the data points of each class. It uses 2 dataset to prove this idea, giving multiple conclusions about the importance of the classes in the visualization of the dataset.

Furthermore, this work define some challenges and future work to develop, using different NLDR techniques and improving this work to become a data analysis tool trying to solve real problems to data analysts that use NLDR.

Table of contents

List of figures

Nomenclature

Acronyms / Abbreviations

AI Artificial Intelligence

CRISP-DM Cross-Industry Standard Process for Data Mining

DM Data Mining

MLP Multi-Layer Perceptron

RBF Radial Basis Function

GT Grand Tour

GTM Generative Topographic Mapping

LDA Linear Discriminant Analysis

ML Machine Learning

NLDR Non-Linear Dimensionality Reduction

PCA Principal Component Analysis

SOM Self-Organizing Maps

Chapter 1

Introduction

1.1 Motivation

Data visualization is an important tool in Artificial Intelligence (AI) in general and Machine Learning (ML) in particular, where it is mostly associated to linear and nonlinear processes of dimensionality reduction. One of the reasons explaining its increasing popularity and wide use is because it helps in data analysis by shedding light on the main characteristics and the behaviour of multivariate data, thus allowing data exploration. In Data Mining (DM), the process of understanding the data under analysis is an extremely important and basic phase. More specifically, in the Cross-Industry Standard Process for Data Mining (CRISP-DM) model [8], data visualization becomes a key element of data exploration in its second phase: Data Understanding, as well as a common one in the third phase, namely Data Preparation, as data visualization often requires data manipulation with distortion.

However, it is sometimes not easy at all to visualize the data as a preliminary step to understand them, especially when amounts of data features are at play. It is the increasingly commonplace problem of high (or even very high) data dimensionality. As a result, strategies for dimensionality reduction that allow decreasing the number of features and, therefore, enable data visualization, are often advisable. There are two main approaches to pursue this goal. The first one, feature selection (FS) aims to find a subset of the features using specific criteria of feature relevance that may or may not be of statistical nature. The second one, feature projection, as a subset of the family of feature extraction methods, applies techniques to reduce and project the high-dimensional space into a space with fewer dimensions. Here we find linear techniques such as Principal Components Analysis (PCA) [9], Linear Discriminant Analysis (LDA) [4], and others. And also, of course, non-linear dimensionality reduction (NLDR) techniques like Self-Organizing Maps (SOM) [3], Generative Topographic Mapping (GTM) [1], and others.

Using linear techniques of feature projection could help the analyst to visualize data faithfully, especially if the data consist of few or a moderate amount of features. Nevertheless, the difficulty of this type of analysis often increases when real data are used, because of their complexity and the large amount of features available for analysis. Trying to find a straight and useful explanation of this data is a hard mission. In this situation, the use of NLDR techniques that allow a flexible visualization of data might be advisable, even if such visualization could be less intuitive.

1.2 Goals of the Thesis

The main idea behind the current thesis is the implementation of a grand tour (GT) [6] strategy for a particular NLDR method, namely GTM, allowing an intuitively and interactive form of data visualization. Specifically, this work will implement the GT for GTM in the Python programming language following its original theoretical definition [6], reaching further objectives, and reporting relevant results. This implementation is meant to become the proof of concept that serves as a foundation for future extensions to other NLDR methods and models.

The Python implementation is developed from scratch, and the GT for GTM is meant to provide the analyst with class-discriminant perspectives of the feature projection, increasing the intuitiveness of the original GTM algorithm. For this purpose, several results using real data and its behaviour are provided and a new option of visualization will be offered to data analysts, allowing the final user, who needs practical results, to decide which tool has best visualization behaviour.

Chapter 2

Methods and Materials

The current chapter provides summary descriptions of some of the techniques and models that are at the basis of the work of these thesis. This is followed by a description of the Python implementation of the interactive visualization method and an account of the datasets with which it has been tested.

The GT-GTM can be described from three different viewpoints. First, the GTM itself is a generative model of the mixture of distributions family that was originally proposed as probabilistic counterpart to the well-know Self-Organizing Maps (SOM). As such, it can be seen as an NLDR model one of whose most interesting characteristics is visualizing high-dimensional data in low-dimensional spaces.

The second viewpoint is that in which the basic GTM is mostly an unsupervised learning technique. Nevertheless, in many instances of data analysis we may want to visualize data in exploratory mode even if data are labeled (and could, therefore, be analyzed using supervised learning approaches). One way to make use of class labels even if in unsupervised mode is by integrating scatter matrices in the GTM.

Finally, multivariate data visualization often aims at providing intuitive views of data representations that are as useful to the analyst as possible. This often means making the visualization at least partially interactive so that the user can control aspects of these views. The GT is one type of such solutions that was also recently proposed to "travel the GTM manifold" in search of maximally discriminant data views.

2.1 Generative Topographic Mapping

As stated in the introduction to this chapter, GTM is a generative ML model that was originally proposed as probabilistic counterpart to the well-know and very widely used Self-Organizing Maps (SOM). Its probabilistic sound foundations can also allow us to understand

it as a constrained variant of the mixture of distributions family of models (in its most basic form, it could be seen, for instance, as a constrained mixture of Gaussians). The constraint has to do with the fact that the centers of this distributions are forced to lay in such a way that they conform a discrete sampling of a smooth low-dimensional manifold. As a result, the GTM is also a manifold learning model. The fact that such manifold can be defined to be 1- 2- or 3-dimensional opens the possibility of using the GTM for straightforward data visualization with far more flexibility than SOM. From this viewpoint, thus, the GTM is a visualization-oriented NLDR ML model.

Unlike SOM, it optimizes a proper cost function (using the same methods that could also be used for general mixture of distributions models) that is proven to converge (at least to a local minimum) and, unlike SOM, it does not require an heuristically shrinking neighborhood, or a decreasing step size.

It is a generative model in the sense that the observed data are assumed to be generated from an unobservable, latent space. This latent space is conveniently sampled for computational expediency and each of the sampled latent points univocally defines a center of a probability distribution in the observed high-dimensional input space (via a smooth function), then adding a noise model in that space. This is not unlike other latent models such as nonlinear factor analysis.

The parameters of the low-dimensional probability distribution, the smooth mapping and the characteristics of the noise are all learned from the training data, usually through the expectation-maximization (EM) algorithm. The approach is also strongly related to density networks, which use importance sampling and a multi-layer perceptron (MLP) to form a non-linear latent variable model. In the GTM, the latent space is a discrete grid of points which, as previously mentioned is projected into observed data space. A Gaussian noise assumption is then made in data space so that the model may become a constrained mixture of Gaussians (although it can take other forms if using different distributions such as t-Student or Bernouilli's).

In theory, an arbitrary nonlinear parametric distortion could be used. The optimal parameters could be found by gradient descent, etc. The commonly suggested approach to the nonlinear mapping is to use a radial basis function network (RBF) to create a nonlinear mapping between the latent space and the observed data space. The nodes of the RBF network then form a feature space and the nonlinear mapping can then be taken as a linear transform of this feature space.

Given a data set of dimension D, $t = (t_1, ..., t_D)$, GTM, as a latent variable model, attempts to estimate the distribution $p(t)$ in terms of a number L of latent variables $x = [x_1, ..., x_L]$. In turn, as a generative model, GTM generates centers of distributions in the observed data

space through a nonlinear mapping function $y(x;W)$, where the nonlinearity is commonly introduced in the form of radially-symmetric Gaussians and where the aforementioned centers can be seen as a discrete sampling of a smooth manifold. Such mapping is mediated by W, a matrix of adaptive parameters that specifies the mapping of x to the points y in the observed data space. Since $p(x)$ can be seen as a distribution of prior probabilities in the latent space, $p(y|W)$ would be its corresponding distribution of probabilities in the space of the data. In this way, the probability of the data given the latent variables and the model parameters can be defined as: [1]

$$p(t|x,W,\beta) = \left(\frac{\beta}{2\pi}\right)^{\frac{D}{2}} exp\left\{-\frac{\beta}{2}\left\|y(x;W)-t\right\|^2\right\}. \tag{2.1}$$

As in other latent variable models, we must integrate the latent variables out in order to estimate the data distribution given the model parameters, in the form:

$$p(t|W,\beta) = \int p(t|x,W,\beta)p(x)dx. \tag{2.2}$$

Now, the adaptive parameters W and β (the inverse variance of the noise model) can be estimated by maximizing the log-likelihood of the model, described as:

$$L(W,\beta) = ln\prod_{n=1}^{N} p(t_n|W,\beta). \tag{2.3}$$

In this scenario, we need to define all components of the previous formula explictly. The prior distribution of latent points in the latent space $p(x)$, due to its discrete sampling nature, can be defined as follows:

$$p(x) = \frac{1}{K}\sum_{i=1}^{K}\delta(x-x_i). \tag{2.4}$$

Then,

$$p(t|W,\beta) = \frac{1}{K}\sum_{i=1}^{K} p(t|x_i,W,\beta). \tag{2.5}$$

and, consequently, the the log-likelihood takes the following simplified form:

$$L(W,\beta) = \sum_{n=1}^{N} ln \left\{ \frac{1}{K} \sum_{i=1}^{K} p(t_n|x_i,W,\beta) \right\}. \tag{2.6}$$

which can easily be optimized using, for instance, the expectation maximization (EM) algorithm.

2.2 Cluster-based visualization with scatter matrices

This work allows us to recalculate the distribution of the data points in the manifold depending on which node of the network in which we are, optimizing the discrimination between classes.

In this way we can travel through the map and examine how they would see the points distributed from that perspective. It allows us to group the tips by decomposing the co variance matrix as the variance of the data within the cluster and the variance of the data between each cluster, as defined in the following formulas: [5]

$$S_T = S_W + S_B \tag{2.7}$$

$$S_T = \sum_{i=1}^{N} ((X_i - m)^T \cdot (X_i - m)), \tag{2.8}$$

$$S_W = \sum_{j=1}^{N_c} \sum_{i=1}^{N_j} ((X_i - m_j)^T \cdot (X_i - m_j)), \tag{2.9}$$

$$S_B = \sum_{i=1}^{N_c} (N_j \cdot (m_j - m)^T \cdot (m_j - m)), \tag{2.10}$$

the matrix of invariant dispersion would be:

$$M = S_W^{-1} \cdot S_B \tag{2.11}$$

and the separation index of the invariant cluster would be:

$$J = tr(S_W^{-1} \cdot S_B). \tag{2.12}$$

Due to these simple equations we can optimize the separation between classes and improve the concentration of points of each class. [5]

Scatter Matrix provides important advantages over data visualization, it is possible to see the data over every point of view using the whole combinations between the characteristics of the data. If it joins Generative Topographic Mapping (GTM), it creates a powerful tool to visualize data. It is helpful, in the data analysis process, because, seeing the transformed data in GTM since every point of view, it makes easy to understand the data behaviour, and furthermore to define better what kind of process apply over data.

2.3 The GTM manifold Grand Tour

In the Manifold Grand Tour [6] arises the first idea of a grand tour through a GTM structure. It is possible to see few advantages in 2 examples, but it is necessary an implementation of the idea, and that is one of the motivations of the development of this work. There is even a procedure to be followed, which would be a guide for this work.

2.4 Methodological process

To deliver on the goals of this thesis, all the reported work has been developed from scratch as Python code. Which get a dataset as input, it is projected with GTM and in this resultant projection is applied scatter matrix process. With the resultant images of scatter matrix, it is made a video, which helps the data annalist to see the whole perspectives of the data and to decide which perspectives could help in any data process, for example to understand the data better.

The dataset is first modeled using the GTM manifold learning method. Then, scatter matrices are used in order to obtain maximally class-discriminatory visualization constrained to the topology of the GTM manifold. This provide as with a number of different *data perspectives* that can be used to build and interactive *tour* that can, for instance, be summarized as a video *travelling* through the model manifold.

The steps of the process can summarily be listed as follows:

- First, data are modeled using the GTM as a manifold learning tool for dimensionality reduction oriented towards data visualization. The GTM modeling requires the a priori definition of a representational discrete grid of points in the model latent space.
- This structure is orthonormalized.
- The different components of the scatter matrices are calculated at each point of the GTM grid.
- We obtain a third direction for data representation along the model manifold.
- Once all different data perspectives along the manifold are calculated, the user could interactively decide how to realize a Grand Tour through the representation space. For illustration, this tour may be represented as a video, but in full-blown implementation of the technique, this tour could become fully interactive, with a user making decisions about where in the manifold to travel that results in the most interesting and/or useful views.

2.4.1 Python implementation

Python is an interpreted high-level programming language that, arguably, has become one the main choices for developing ML and DM tools, due to its versatility and simplicity. [7] In this thesis, the following libraries have been used:

- `pandas`: The pandas library is the most popular library to work with data in this work is really useful for matrix operations.
- `numpy`: This library helps to numerical and mathematical operations.
- `pyplot` and `mplot3d` : This libraries are the main tools for graphical representation and visualization of the data.
- `os`: The OS library is the one used to work with directories and files creation.
- `itertools`: This library allows generate permutations and combinations in an easy way.
- `cv2`: The cv2 library is the one used to create videos from a given set of images.
- `scipy`: This library in this project allows to calculate the eigenvectors and eigenvalues and to calculate thee distance matrix.

For further information about the code developed in this paper, we can find the most relevant functions detailed in the appendices to the thesis. Moreover, the complete code used in the whole project has been uploaded to the Github portal. [2]

2.4.2 Output

The result of the procedure previously detailed is a collection of maximally class-discriminant 3-D views of the data under analysis. There is one view for each of the nodes (points of the GTM representation latent grid) of the model. Given that these grids are often chosen to be square (unless there is any prior or contextual information that might lead to alternative choices of grid shape), a 10×10, for instance, would generate 100 data views that could be navigated according to any feasible criterion. By default, and as previously said, a video with a pre-established tour could be generated. Alternatively, though, the data analyst might be most interested in investigating, for instance, those areas of the map in which data have more densely been mapped, or those areas with any particular practical interest from the point of view of a given application.

2.5 Materials

Originally, this thesis was meant to illustrate the usefulness of the implemented methods using an original dataset including information about electricity consumption in a consumer market. Unfortunately, these data were not made available to us and, instead, we carried on with the illustration using the following datasets. They all belong to the well-known UCI machine learning repository. Only subsets including class labels and real-valued features were selected for analysis.

2.5.1 Iris Dataset

The Iris dataset is, arguably, the best-known benchmark low-dimensional dataset in ML and Pattern Recognition, due to their clearly defined characteristics, speacially those related to class distribution.

These are data resulting to the recollection of information on morphologic variation of the Iris flower variety. Data belong to three different varieties of the flower (50 of each variety/class), namely Iris setosa, Iris virginica and Iris versicolor. Four measurements are informed in the data: petal lengths and widths, and sepal lengths and widths.

This dataset in particular is perfectly suited to the illustration of the proposed visualization method, as it complies with the aforementioned requirements.

2.5.2 Connectionist Bench Dataset

This dataset is a contribution by Terry Sejnowski at the Salk Institute and the University of California. It was developed in collaboration with Paul Gorman.

It includes a collection of bouncing sonar signals off metal cylinders and rocks; 111 patterns for mines (metal cylinder) at different angles (up to 90 degrees) and 97 patterns for rocks from different angles up to 180 degrees.

The 60 measured parameters represent the energy in a particular signal frequency band. The values taken by each of these parameters are in the range between 0 and 1. There is a column that identifies the class, so that *M* represents the mines and *R* the rocks. This column is converted into a numerical binary labels, so that *R* becomes 0 and *M* becomes 1.

Chapter 3

Experiments: Results and Discussion

Following the process defined in the previous section, we carried out some experiments in order to compare the different forms of visualization of the data. First of all, we tested different GTM latent representation grid sizes. Specifically, we tested square grids of size 7, 10, 15 and 20 nodes per component (that is, from 49 to 400 latent points or GTM nodes). Obviously, the greater the size of the grid, the more computational processing capacity it requires. Grids of arbitrary sizes could be considered, but they are not required for the illustration of the GT for GTM process.

The first data visualizations extracted in each experiment correspond to the straightforward use of the standard GTM model, using the class labels *a posteriori*, that is, not including them in any part of the model training process. For illustration, a total number of 8 data visualization images were produced for each dataset in this category of primary images. Subsequently, the GT GTM procedure defined in the previous section is executed, obtaining a total number of images that is equal to the number of latent points in the GTM grid (for instance, 49 images for a 7×7 square grid, and so on). From these images, we will pick up only some of the most representative to illustrate the contrasting behavior from different points of the grid with respect to a primitive sample of data with the standard GTM.

3.1 Results

In this section we will see the results obtained when viewing the data after applying the GTM algorithm alone and without identifying the classes. We will also show how the data would be displayed if we identify the classes. And finally we will visualize the data some points of view after applying MGT to the data identifying the classes

3.1.1 Iris Dataset

GTM with no labelled classes

Here we will analyze the visualization of the data of the 4 three-dimensional images, 1 for each case according to the grid number: 7.10, 15, 20; after transforming the data with GTM. These data will not be identified according to their classes, and so we will analyze if there is any separation in the classes that we can intuitively find. Also we will try to see the influence of the number of nodes per component in the GTM structure.

In the Figure3.1 we can see a separation a little to the left of the central zone of component 1, approximately at the level of -0.75 and -0.5, which could separate the classes, however it would only have this In this way, a visual separation of the data in the GTM matrix, in general, the data of points very distributed in all the nodes of the structure are seen. With respect to component 3, a separation can be seen in the lower area at the height of -0.75 and 0.00 above all in the -1.00 and -0.25 part of component 2. With respect to component 2, many separations are not distinguished.

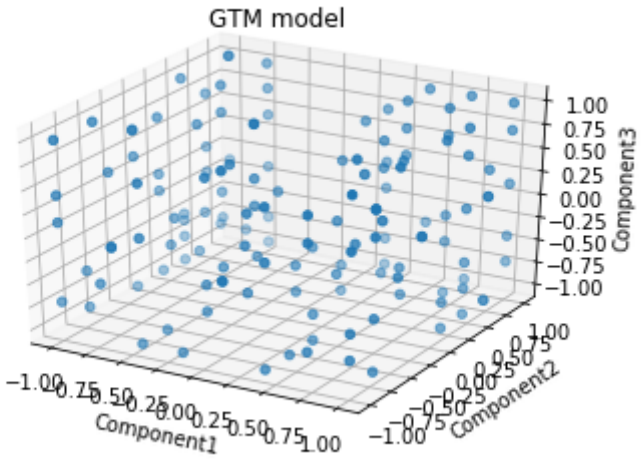

Fig. 3.1 The 3D scatter plot of the Iris dataset with a GTM structure of 7 nodes per component.

In Figure 3.2 we can see a separation quite similar to the figure 3.1 but a little wider at the top and bottom of component 3 above all. What could lead us to conclude that in that area there would be a division between classes. With respect to components 2 and 3 can not be detailed greater changes than in the figure 3.1

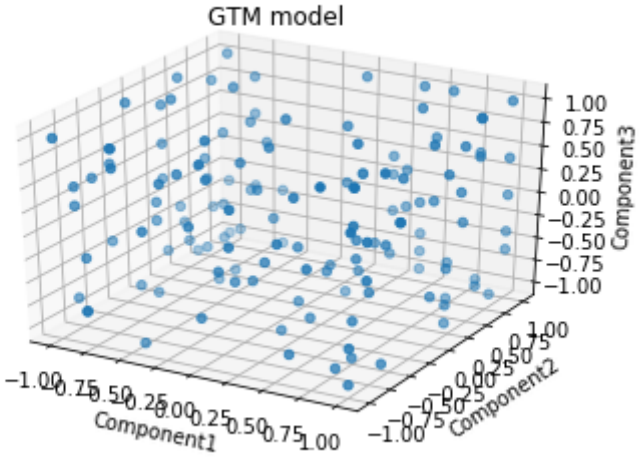

Fig. 3.2 The 3D scatter plot of the Iris dataset with a GTM structure of 10 nodes per component.

In Figure 3.3 we see a behavior similar to Figure 3.2, in the upper part of component 3, but some data points are seen in the lower part of component 3 They would be in the place where there is a marked division between classes. Even so, a separation can be seen in the same area as the figures 3.1 and 3.2. With respect to components 2 and 3 can not be detailed greater changes than in the figure 3.1

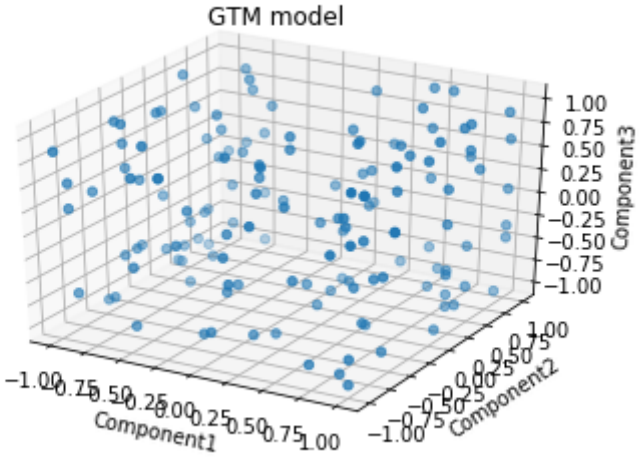

Fig. 3.3 The 3D scatter plot of the Iris dataset with a GTM structure of 15 nodes per component.

In Figure3.4 You see the data a little more "tight" but the separation seen in the previous images remains in all the components.

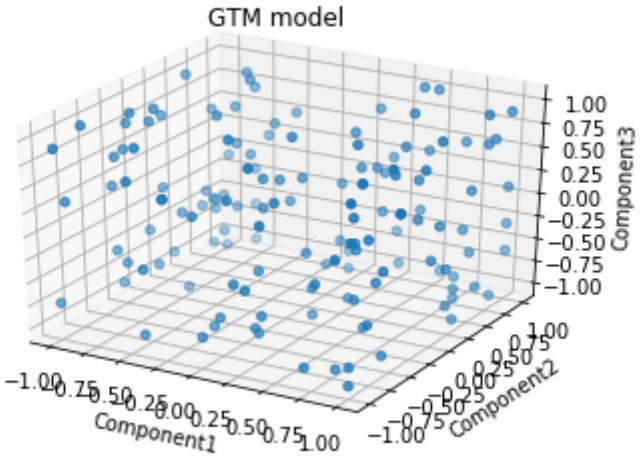

Fig. 3.4 The 3D scatter plot of the Iris dataset with a GTM structure of 20 nodes per component.

GTM with labelled the classes

In this section we will be able to decipher which were the classes of the 4 images analyzed in the previous section, so that we can discover the distribution of the points of each class in the GTM structure.

After having made a brief analysis a bit intuitive from the behavior of the data after having been converted to GTM but not having signaled their classes, we will see a bit the behavior of the data with their respective classes and then contrast it with the MGT.

In Figure 3.5 we can see that there is a marked separation between the setose iris class, with respect to the other 2 classes, iris versicolor and iris virginica. However, there is no such clear division between iris versicolor and iris virginica classes. It is noticeable that the green dots that represent the iris versicolor distribution are more to the right of component 1 tending to 1, while the red dots that represent the iris virginica distribution, are closer to the large division with the class of the blue dots that represent the distribution of the iris setosa class.

It is also worth mentioning that, despite the great division of the iris setosa class with respect to the other 2 classes, there are points like iris versicolor outliers and iris virginica near the iris setosa class data points.

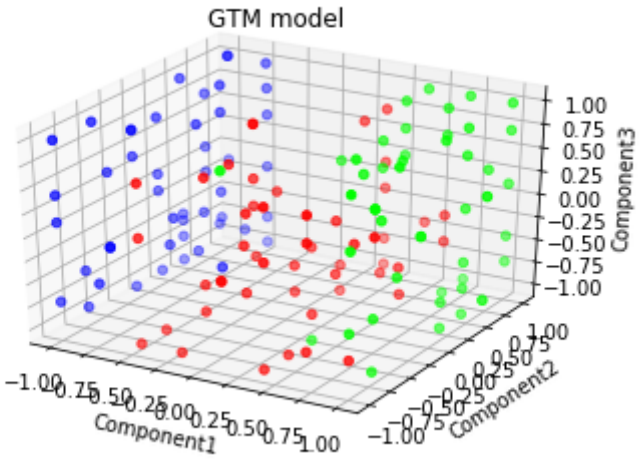

Fig. 3.5 Labelled 3D scatter plot of the Iris dataset with a GTM structure of 7 nodes per component. The blue dots represent the distribution of the Iris Setosa, the red dots represent the distribution of the Iris Virginica and the green dots represent the distribution of the Iris Versicolor.

In Figure 3.6 the separation that had been maintained and we had discussed it comparing the figures 3.1 and 3.2, it can be noticed yet, instead of improving the separation of the iris setosa class (blue dots) it does not happen, rather it is seen as if the red dots of the iris class virginica approached towards the iris setosa class, being noticed less to separation of the iris setosa class.

As for the other classes, the same behavior is maintained, that in the figure 3.5 that is to say, it is not noticed as much, nevertheless the green points that represent the iris versicolor class is kept to the right of component 1 and the red dots of the iris virginica class are close to the division that was in 3.5 and they even get to mix a little with the iris setosa class.

Regarding the outliers of the classes iris versicolor and iris virginica commented in the figure 3.5 are still present in this representation of the data.

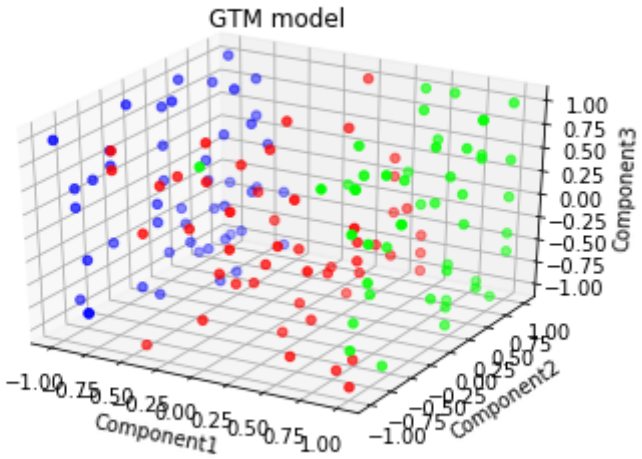

Fig. 3.6 Labelled 3D scatter plot of the Iris dataset with a GTM structure of 10 nodes per component. The blue dots represent the distribution of the Iris Setosa, the red dots represent the distribution of the Iris Virginica and the green dots represent the distribution of the Iris Versicolor.

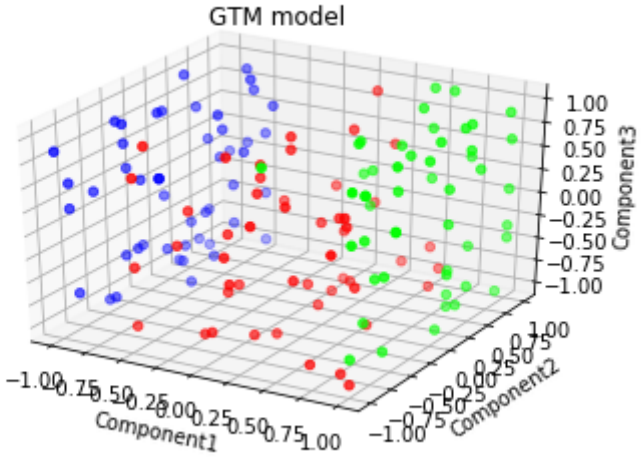

Fig. 3.7 Labelled 3D scatter plot of the Iris dataset with a GTM structure of 15 nodes per component. The blue dots represent the distribution of the Iris Setosa, the red dots represent the distribution of the Iris Virginica and the green dots represent the distribution of the Iris Versicolor.

The Figure 3.7 shows us a behavior similar to the image 3.6 with respect to the separation of the classes, perhaps the most interesting thing to detail in this image is that the outliers of iris versicolor, especially the outlier of iris virginica are separated from the points of the iris setosa class.

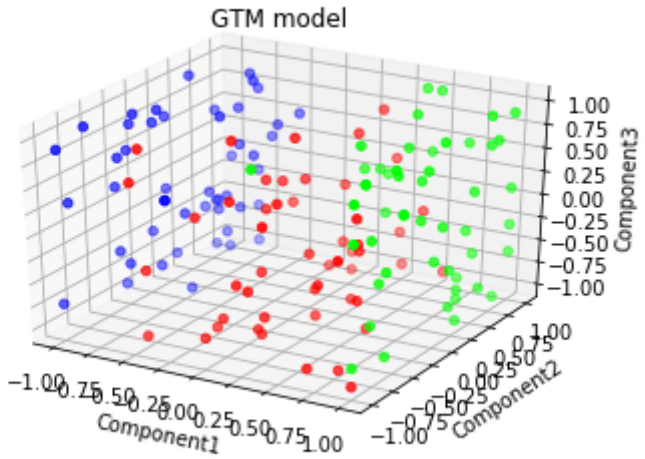

Fig. 3.8 Labelled 3D scatter plot of the Iris dataset with a GTM structure of 20 nodes per component. The blue dots represent the distribution of the Iris Setosa, the red dots represent the distribution of the Iris Virginica and the green dots represent the distribution of the Iris Versicolor.

In Figure 3.8 we have a behavior practically similar to the figure 3.7 having the iris setosa class (blue dots) to the left of component 1 very grouped together. While the iris versicolor class (green dots) are grouped rather to the right of component 1, although it has an outlier between the iris virginica class (red dots) that are rather, in the center of component 1 filling in the space that would exist between the iris setosa and iris versicolor classes.

MGT

Below you will see the most relevant images of the Manifold grand tour where you have traveled through a manifold of 15 nodes per component. The resultant images do not vary with respect to the manifold size from the 15 nodes per component. So we will only work with a manifold of 15 nodes per component.

In Figure 3.9 you can see how the data points are diagonalized on the component 2 and component 3 planes from -2 for component 2 and 2 for component 3 to 2 for component 2 and - 2 for component 3. The points closest to component 1 would be those corresponding

to the iris versicolor class (green dots) following the iris virginica class (red dots) that can be seen after the green dots. Finally, a few blue dots corresponding to the iris setosa class after the red dots are visualized. In general, from the perspective of the current point of the manifold, we can not detail more relevant information.

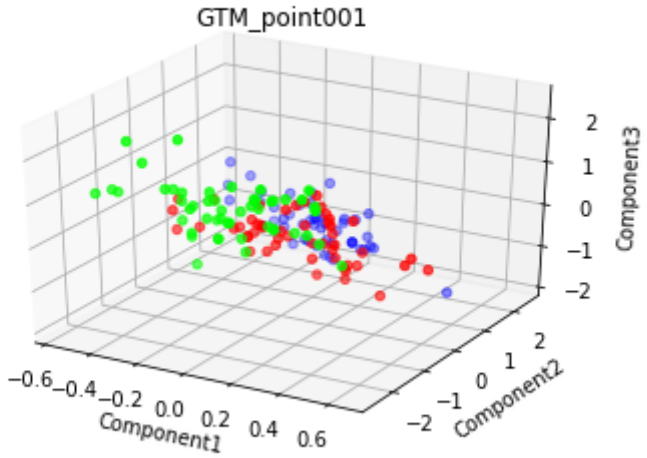

Fig. 3.9 Scatter plot of the Iris dataset seen since the point 1 of the GTM structure. The blue dots represent the distribution of the Iris Setosa, the red dots represent the distribution of the Iris Virginica and the green dots represent the distribution of the Iris Versicolor.

The Figure 3.10 represents the perspective of point 35 of the manifold of 15 nodes per component. Where you can see a clear separation of the data with respect to the image 3.9 because here something similar to a diagonal is shown in component 2 and component 3 from -2 in component 2 and -2 in component 3 to 2 in component 2 and 2 in component 3. So the iris versicolor class (green dots) are closer to component 1. Then the data points corresponding to the iris virginica class that are found below 0 for component 3 and after 1 for component 2 are clearly grouped together and distant from the other 2 classes. Finally, like the other 2 classes, the iris setosa class is grouped after point 0 in component 3 and after point 1 in component 2, practically further away than component 1. This perspective shows a clear distribution of the data with respect to the perspective of the image3.9.

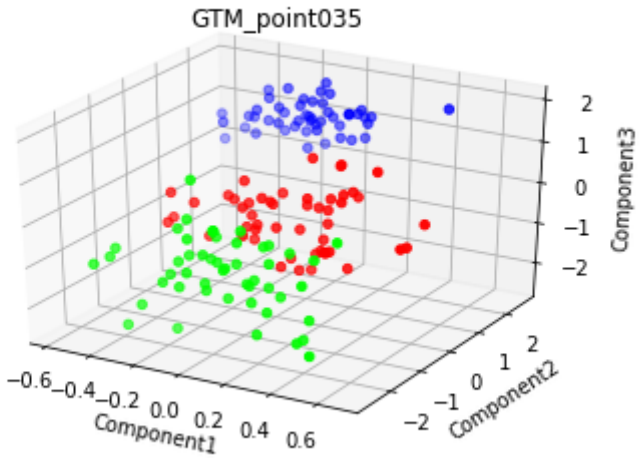

Fig. 3.10 Scatter plot of the Iris dataset seen since the point 35 of the GTM structure. The blue dots represent the distribution of the Iris Setosa, the red dots represent the distribution of the Iris Virginica and the green dots represent the distribution of the Iris Versicolor.

3.1.2 Connectionist Bench Dataset

GTM with no labelled classes

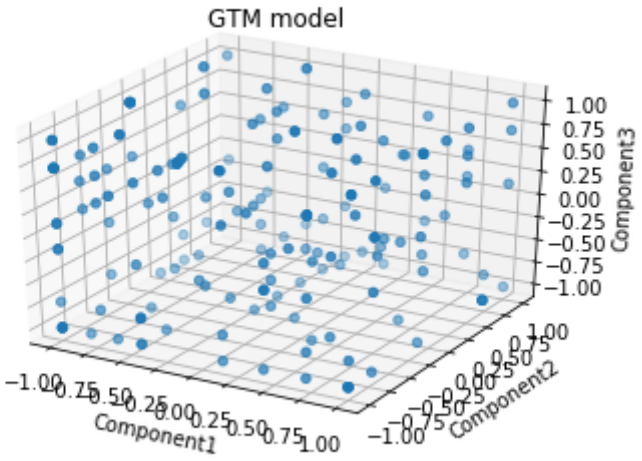

Fig. 3.11 The 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 7 nodes per component.

In Figure 3.11 you see a great expansion of the data points through the whole image, with very few spaces where the separation of the classes could be noticed. There are some small spaces between some data points in component 3 approximately at a height of -0.75 and -0.5.

In Figure 3.12 you can see a behavior very similar to the figure 3.11 however you can see a diagonal entry to the height -0.75 and -0.25 of the component 2 that would continue more or less to -0.75 of the component 1 of diagonal way.

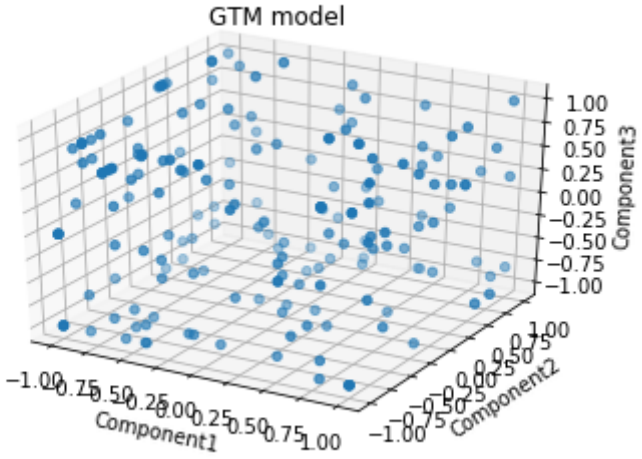

Fig. 3.12 The 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 10 nodes per component.

The Figure 3.13 shows a behavior very similar to the figure 3.11 with few spaces that can be seen a division of data. The only detail that we can see is that the space that is in the height -0.5 and -0.25 in the component 3, is a little higher than the view in the figure 3.11 that is -0.75 and -0.5 of component 3.

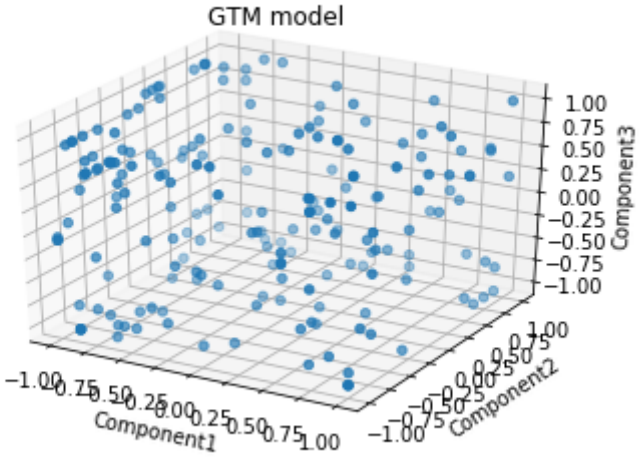

Fig. 3.13 The 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 15 nodes per component.

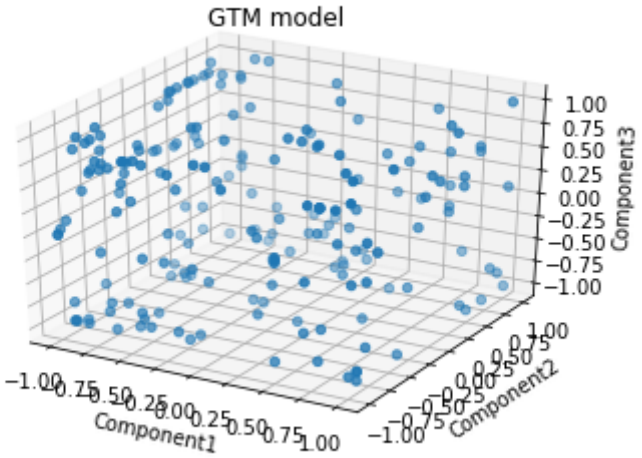

Fig. 3.14 The 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 20 nodes per component.

In Figure 3.14 with respect to the generated distribution of the points is quite similar to the previous figures 3.11, 3.12 and 3.13. In this way the space in component 3 is still visible in the approximate height between -0.75 and -0.25, however an input is also displayed that could be a division of classes in the height of 0.00 and 0.25 of component 1 that crosses a little irregularly all the component 2 up to the height -0.75 and -0.50 of the component 1.

GTM with labelled classes

In this section we compare the GTM images previously seen without the identification of the classes and what is the result after assigning the corresponding classes to the datapoints.

In the Figure 3.15 you can see the green points corresponding to the mines very grouped in zone -1 and -0.75 of component 2 to -0.5 of a component 3, but there are still green points of the class mines distributed in many nodes of the GTM structure. something similar to the rocks class, distributing through the GTM structure without finishing defining a clear separation of the classes. Finally, the separations of points found in the figure 3.11 do not represent a real division of the classes.

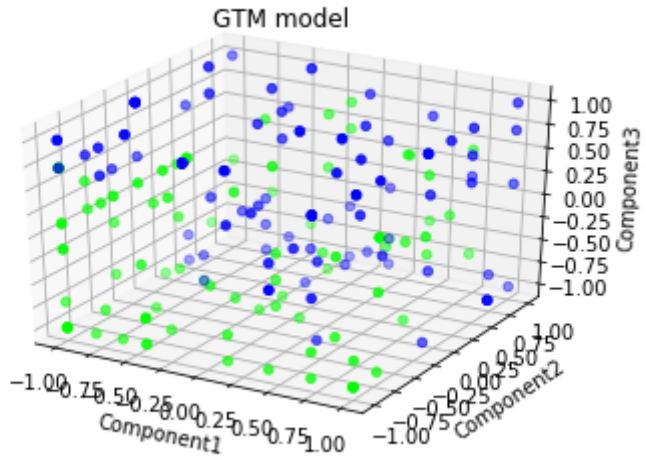

Fig. 3.15 Labelled 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 7 nodes per component. The green dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

In Figure 3.16 something very similar to that described in the figure 3.15 is seen, the data points of the mines and rocks are very distributed among the GTM structure without finally reaching a significant separation between classes.

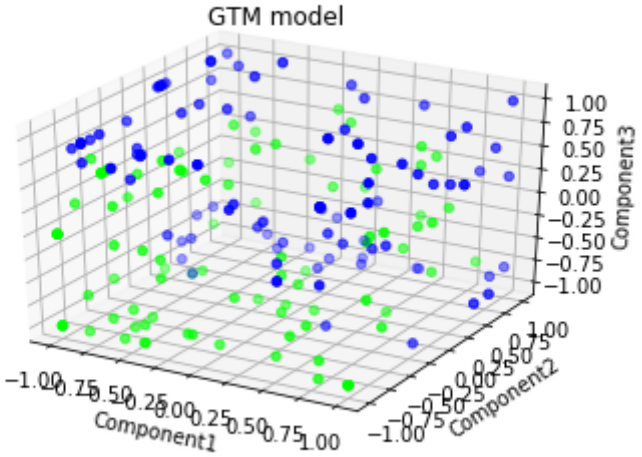

Fig. 3.16 Labelled 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 10 nodes per component. The green dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

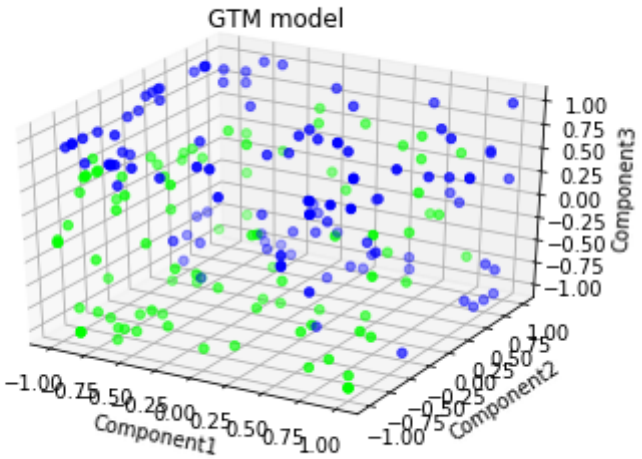

Fig. 3.17 Labelled 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 15 nodes per component. The red dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

In the Figure 3.17 even though the data of the class mines are kept between -1.00 and 0 of the component 2 through the component 1 from -1 to 0.5 of the component 3, they are

not finished. group and separate from the class rocks (blue points). So finally we can see a behavior similar to that of the figures 3.15 and 3.16.

In Figure 3.18 the distribution of the data points of the mines and rocks has a behavior quite similar to the one detailed in the figure 3.17, that is, a section that could differentiate the points green corresponding to the class of mines, but in the end do not finish grouping the entire class and separate it from the blue points corresponding to the class of rocks. Comparing this image with its similar without marking the classes (figure 3.14), shows us that despite having a space between marked data points, does not mean that it is finally a class separator space.

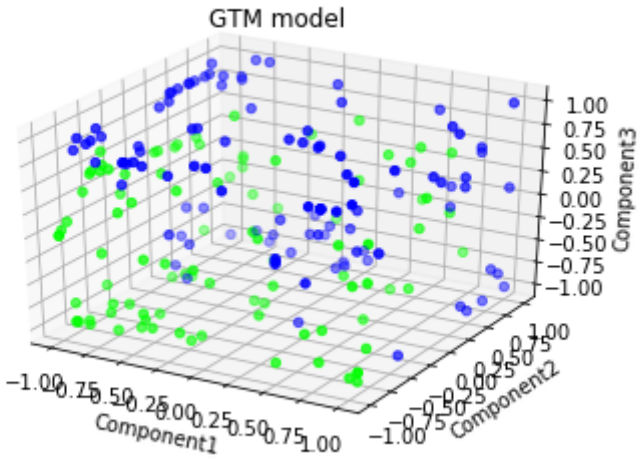

Fig. 3.18 Labelled 3D scatter plot of the Connectionist Bench dataset with a GTM structure of 20 nodes per component. The green dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

MGT

Now the data behavior of the Dataset Connectionist dataset will be analyzed after the grand tour through a GTM structure with 15 nodes per component.

In Figure 3.19 the perspective of node 1 of the GTM structure is seen and a large condensation of data is seen between 0 and 1 for a component 2, -1 and 0.25 for component 1 and -0.6 and 0.6 for component 3. Apparently a large concentration of data between classes, however do not end up visually separate classes.

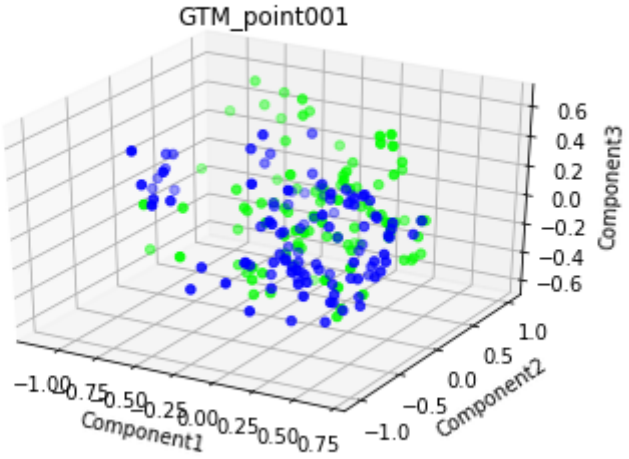

Fig. 3.19 Scatter plot of the Connectionist dataset seen since the point 1 of the GTM structure. The green dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

The Figure 3.20 corresponding to the visual representation of the node 11 perspective of the GTM structure shows that the data is not as concentrated between classes as in the figure 3.19 and are rather dispersed with respect to the perspective figure of node 1 of the GTM structure.

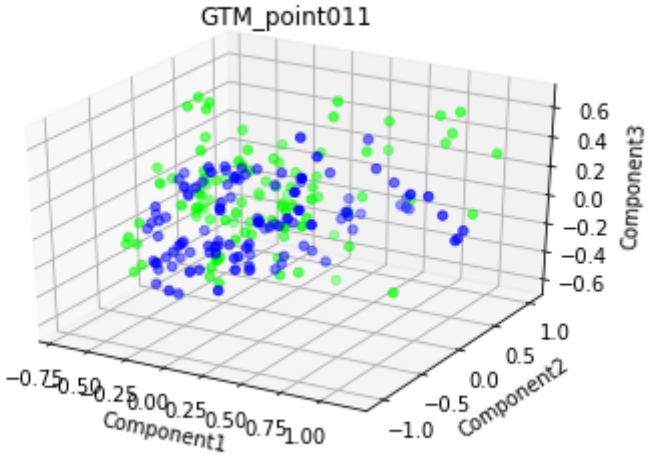

Fig. 3.20 Scatter plot of the Connectionist dataset seen since the point 11 of the GTM structure. The green dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

In Figure 3.21 we see the distribution of points from the perspective of node 39 of the GTM structure. In this figure we can appreciate the data of both classes, both rocks and mines, scattered in the image, without concentrating on any particular section, in the same way, there is no separation between the classes marked. Comparing it with the image of the previous perspective, the figures 3.20, the points of the data are distributed in a different way, but they continue with the same pattern.

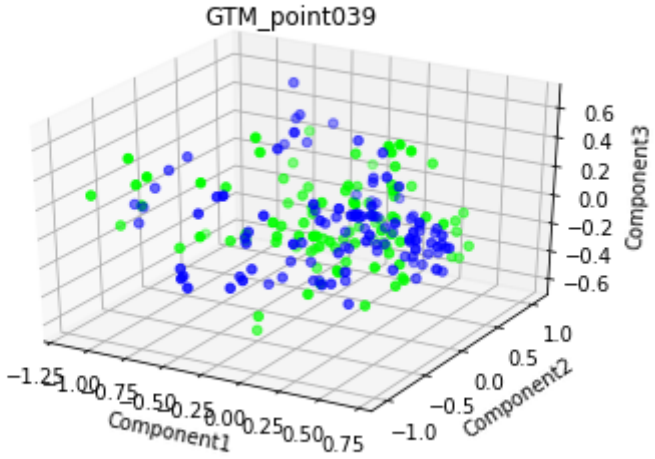

Fig. 3.21 Scatter plot of the Connectionist dataset seen since the point 39 of the GTM structure. The green dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

The Figure 3.22 shows the distribution of the points of the Connectionist bench dataset from the perspective of node 126 of the GTM structure. Where we can see a behavior similar to the figure 3.19 from the perspective of node 1 of the GTM structure. That is, there is a concentration of blue dot data corresponding to the rocks class, however it also has some scattered data points of this concentration. And there is also a concentration of green points corresponding to the data of the mines, but without finally having a well-defined separation between classes, making it difficult to visualize the data well separated by classes.

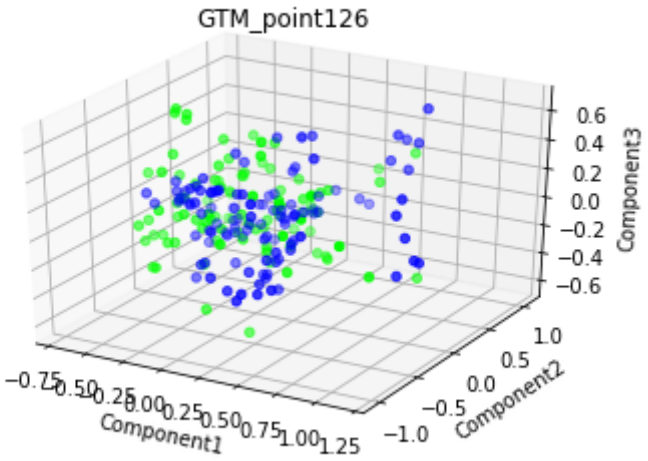

Fig. 3.22 Scatter plot of the Connectionist dataset seen since the point 126 of the GTM structure. The green dots represent the distribution of the mines and the blue dots represent the distribution of the rocks.

3.2 Discussion

In this section, we will compare the results of the data shown applying just the GTM algorithm, both identifying the classes of the data and without identifying the classes. We will contrast these results with the MGT procedure so that we can check the behavior of the data from the purely GTM vision against to the vision after traveling through the manifold points.

3.2.1 GTM colored-classes vs standalone GTM

The first comparison will be to see the data transformed into GTM and to construe the conclusions that could be reached by looking at the data without identifying the classes and what happens when the classes are placed.

Iris Dataset

Regarding the iris dataset as we mentioned in the section 3.1.1 despite having identified a marked separation in the data without the identified classes (figures 3.1, 3.2, 3.3 and 3.4) it is not possible to identify the outliers near the setose iris class, which are noticed once the classes have been identified. In the figures without identified classes, it is not noticed

that although the iris versicolor and iris iris classes are together, if the data points of each class are kept together (figures 3.5, 3.6, 3.7 and 3.8). Finally, the data points identifying their classes provide more information about the data than without identifying the classes.

Connectionist Bench Dataset

Regarding the Connectionist Bench Dataset looking at the figures 3.11, 3.12, 3.13 and 3.14 without identified classes, some space options are proposed class separators. However, seeing the distribution of the data points in the GTM structures identifying the classes (figures 3.15, 3.16, 3.17 and 3.18), there is no well-defined separation between classes, in fact, the data points of the mine and rock classes are very interconnected.

3.2.2 GTM colored-classes vs MGT

The next comparison will be between the visualization of the data in the GTM structures with identified classes and the most relevant perspectives of the grand tour.

Iris Dataset

In the iris dataset, although the data points in the GTM structures are quite identifiable, there are still some outliers that do not finish grouping with their respective classes. If we compare the figure 3.9 with the different options of figures with the identified classes (figures 3.5, 3.6, 3.7 and 3.8), this perspective definitely does not improve the visualization of the data with respect to the simple use of the GTM algorithm.

However, the figure 3.10 that shows the perspective of the node 35 of the GTM structure, gives us a better distribution of the data, being able to better understand the behavior of the iris classes setosa, iris versicolor and iris virginica, that in the figures with identified classes, they are noticed, in addition it groups better the outliers that when applying the algorithm of GTM does not happen.

Connectionist Bench Dataset

In the Connectionist Bench Dataset when applying the GTM algorithm to the data, identifying the classes, it is not possible to appreciate a well defined separation of the data in all the images of the different GTM structures that we have made in this work (figures 3.15, 3.16, 3.17 and 3.18). However, when viewing the data points from some perspectives of the nodes of the GTM structure, the little separation of the mines and rocks is maintained, although it is

perceived in certain perspectives (figures 3.19 and 3.22) a concentration of the same classes, although with some scattered points.

Chapter 4

Conclusions and future work

4.1 Conclusions

After the analysis made to see the behavior of the Iris dataset and Connectionist Bench as illustrations for the application of the standard GTM algorithm on its own as compared to the use of the grand tour through a GTM structure approach, we can conclude that:

- Data with a significant separability between labeled classes such as the Iris dataset will show a good perspective of the data in the grand tour, improving even the perspective of applying the standard GTM algorithm on its own.
- Data with little separability between data classes, as in the case of Connectionist Bench dataset, will not show a significant improvement in the visualization of the data points, however some perspective could show a condensation of the data, allowing better analysis with respect to the information of the data.
- In general, an intuition of the distribution of the data is shown when carrying out a large tour through a GTM structure with respect to the visualization of the data after applying the GTM algorithm to the data.

4.2 Future work

Some of the challenges and future work that arises after the development of this work are:

- Extend the grand tour visualization approach to other algorithms with LDR and NLDR, such as SOM, among others, and even make a multiple comparative of the grand tour between the different algorithms for the same datasets.

- Develop a suggested guide of relevant perspectives comparing the different perspectives of the nodes and discriminating those images with little difference with respect to a threshold defined by the user.
- Create a user-friendly interactive tool with an intuitive graphical interface, so that the data analyst can decide which path of perspectives she or he considers appropriate according to her or his interests.

References

[1] C. M. Bishop, M. Svensén, C. K. I. W. (1998). Gtm: The generative topographic mapping. *Neural Computing*, 10(1):215–234.

[2] Github Code of this work (2018). Github code of this work. [online] https://github.com/xavieras1/GTGTM.

[3] Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21:1–6.

[4] L. Ming, Y. B. (2005). 2d-lda: A statistical linear discriminant analysis for image. *Pattern Recognition Letters*, 26(5):527–532.

[5] P. J. G. Lisboa, I. O. Ellis, A. R. G. F. A. M. B. D. (2008). Cluster-based visualization with scatter matrices. *Pattern Recognition Letters*, 29:1814–1823.

[6] P. J. G. Lisboa, J. D. Martín-Guerrero, A. V. (2015). Making non linear manifold learning models interpretable: The manifold grand tour. *Expert Systems With Applications*, 42:8982–8988.

[7] Python oficial website (2018). Python oficial website. [online] https://www.python.org/.

[8] Shearer, C. (2000). The crisp-dm model: The new blueprint for data mining. *Data Warehousing*, 5(4):13–22.

[9] Williams, H. A. L. J. (2010). Principal component analysis. *Wires Computational Statistics*, 2(4):433–459.

Appendix A

Python implementation of Grand Tour using GTM

In this appendix, we provide the main parts of the code with the Python implementation of the Grand Tour for GTM with scatter matrices-based discriminant clustering.

Listing A.1 Python code to obtain the scatter matrices used for maximally discriminant cluster-based visualization [5]

```
1
2  def getVariances(data,targets,mean):
3      Sw = 0
4      Sb = 0
5      class_means = pd.DataFrame()
6      pd_target = pd.DataFrame(targets)
7      for current_class in np.unique(targets):
8          current_class_data = ...
               data.iloc[pd_target[pd_target[0]==current_class].index]
9          current_class_mean = np.mean(current_class_data)
10         class_means = ...
               class_means.append(current_class_mean,ignore_index=True)
11         x_i = current_class_data - current_class_mean
12         diff_mean = pd.DataFrame(current_class_mean - mean)
13         Sw += x_i.T.dot(x_i)
14         Sb += diff_mean.dot(diff_mean.T)
15     return Sw, Sb, class_means
```

Listing A.2 Python code involving the iteration over the latent points of the GTM visual representation grid [6]

```
1
2  def GT(data,targets,n_grids=15,folder='./'):
3      initial_point = -1
4      final_point = 1
5      grid_lenght = (final_point-initial_point)/(n_grids-1)
6      GTM_structure = np.array(list(product(np.arange(initial_point, ...
           (final_point+grid_lenght), grid_lenght), repeat=2)))
7
8      GTM_structure_orthonormal, R = np.linalg.qr(GTM_structure)
9      GTM_structure_orthonormal2 = GTM_structure_orthonormal * ...
           GTM_structure_orthonormal
10
11     GTM_structures=[]
12     for point in range(len(GTM_structure_orthonormal2)):
13         newXData = data*GTM_structure_orthonormal2[point,0]
14         newYData = data*GTM_structure_orthonormal2[point,1]
15         newData = data-newXData-newYData
16         GTM_structures.append(pd.DataFrame(newData))
17
18     if not path.exists(folder):
19         makedirs(folder)
20     i=1
21
22     GTM_transformed=[]
23
24     for current_manifold in GTM_structures:
25         overall_mean = np.mean(current_manifold,0)
26         current_data = current_manifold  - overall_mean
27         n_dimensions = len(current_data.columns)
28
29         Sw, Sb, class_means = ...
               getVariances(current_data,targets,overall_mean)
30
31         M = np.linalg.pinv(Sw).dot(Sb)
32         J = np.trace(M)
33
34         print('Original data')
35         print('Cluster performance indices: tr(Sb)= ...
               '+str(np.trace(Sb))+' tr(Sw)= '+str(np.trace(Sw))+' ...
               tr(Sb)/tr(Sw)= '+str(np.trace(Sb)/np.trace(Sw)))
36         print('Invariant criterion:         tr(inv(Sw)*Sb)= '+str(J))
```

```
37
38          vectors_Sw, diag_Sw = eigh(Sw)
39          values_M, vectors_M = eigh(M)
40          sorted_values_M = np.sort(values_M)
41
42          first_eigen = ...
                np.argsort(values_M)[n_dimensions-3:n_dimensions]
43
44          main_vectors = vectors_M[:,firstEigen]
45
46          high_eigen = sorted_values_M[n_dimensions-3:n_dimensions])
47
48          # explaination of 3 first eiginvalues
49          explaination = 100*sum(high_eigen/sum(sorted_values_M)
50
51          print('Explaination '+str(explaination))
52
53          transformed_means = class_means.dot(main_vectors)
54          transformed_data = current_data.dot(main_vectors)
55
56          GTM_transformed.append(transformed_data)
57
58          name="GTM_point"+'{0:03d}'.format(i)
59
60          fig = plt.figure(name)
61          ax = fig.add_subplot(111, projection = '3d')
62          x=transformed_data[0]
63          y=transformed_data[1]
64          z=transformed_data[2]
65          ax.scatter(xs=x,ys=,zs=z,c=targets, cmap=plt.cm.brg)
66          plt.gca().set_title(name)
67          plt.gca().set_xlabel("Component1")
68          plt.gca().set_ylabel("Component2")
69          plt.gca().set_zlabel("Component3")
70          plt.savefig(folder+"/"+name+".png")
71          plt.show()
72
73          transformed_means = np.mean(transformed_data,0)
74
75          T_Sw, T_Sb, t_class_means = ...
                getVariances(transformed_data,targets,transformed_means)
76          T_M = np.linalg.pinv(T_Sw).dot(T_Sb)
77          T_J = np.trace(T_M)
78
```

```
79          print('Original data projected onto 3D using the largest 3 ...
               eigenvalues of the scatter matrices in the original ...
               domain')
80          print('Cluster performance indices: tr(Sb)= ...
               '+str(np.trace(T_Sb))+' tr(Sw)= ...
               '+str(np.trace(T_Sw))+' tr(Sb)/tr(Sw)= ...
               '+str(np.trace(T_Sb)/np.trace(T_Sw)))
81          print('Invariant criterion:         tr(inv(Sw)*Sb)= ...
               '+str(T_J))
82
83          i+=1
```